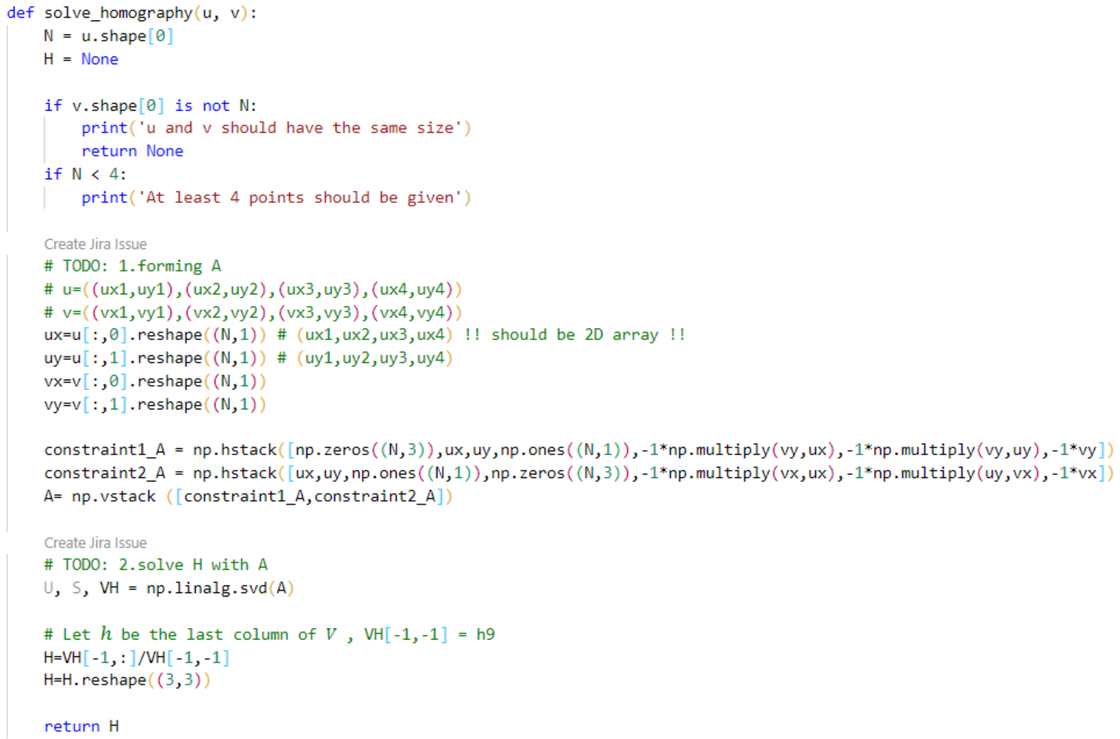
HW3_Report

student ID : b08505024

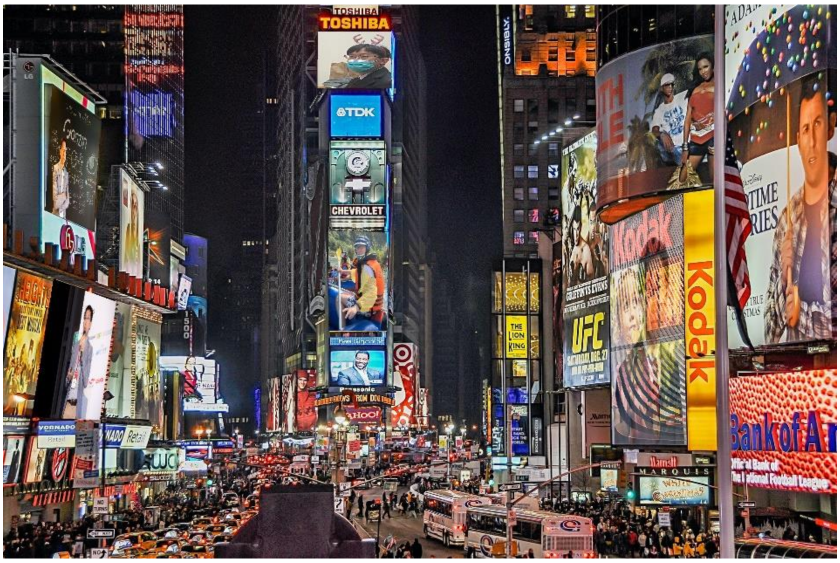
name : 工海三 劉虹伶

Part1: Homography estimation

- Paste the function code solve_homography(u, v) & your warped canvas.

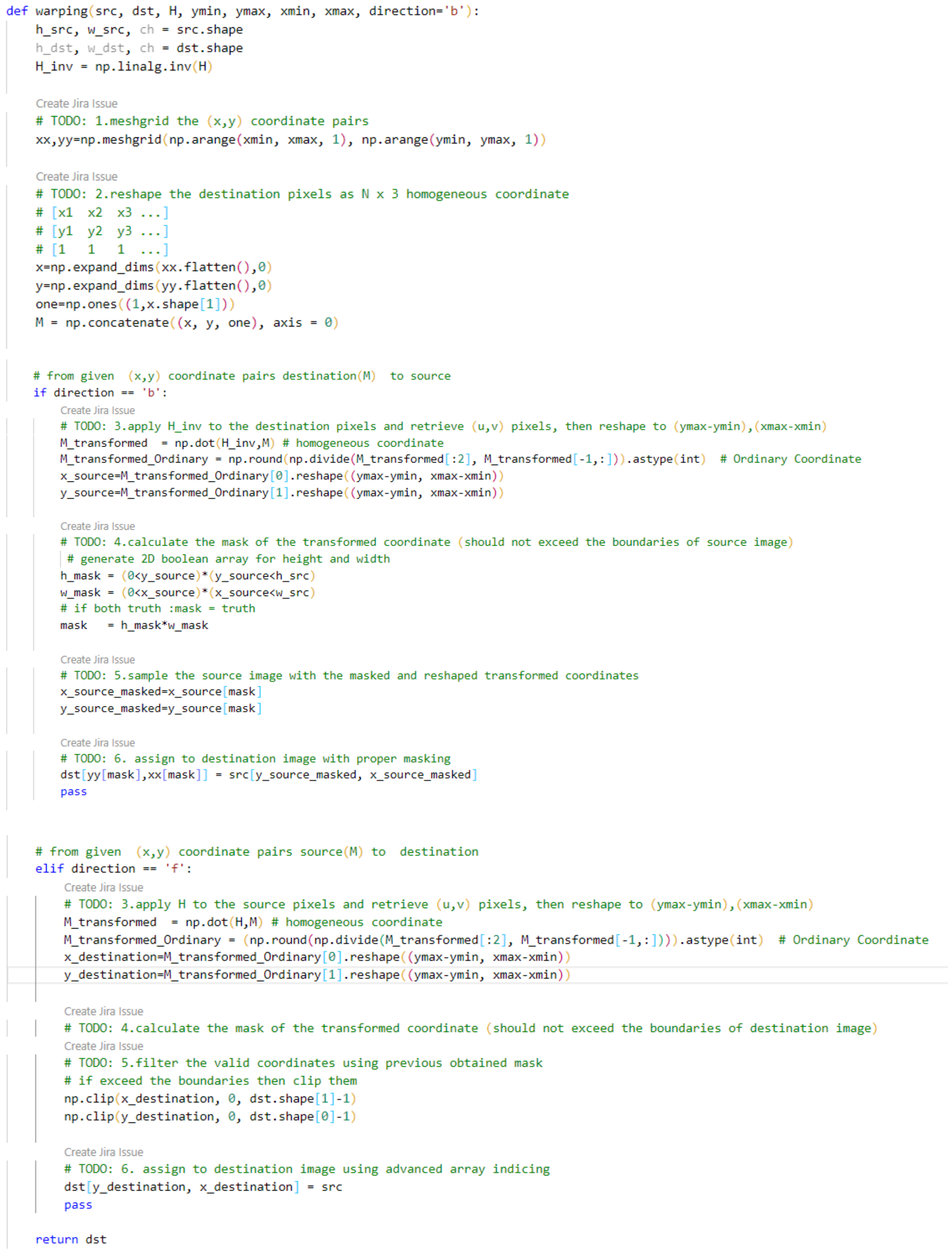


```
def solve_homography(u, v):
    N = u.shape[0]
    H = None

    if v.shape[0] is not N:
        print('u and v should have the same size')
        return None
    if N < 4:
        print('At least 4 points should be given')

    Create Jira Issue
    # TODO: 1.forming A
    # u=((ux1,uy1),(ux2,uy2),(ux3,uy3),(ux4,uy4))
    # v=((vx1,vy1),(vx2,vy2),(vx3,vy3),(vx4,vy4))
    ux=u[:,0].reshape((N,1)) # (ux1,ux2,ux3,ux4) !! should be 2D array !!
    uy=u[:,1].reshape((N,1)) # (uy1,uy2,uy3,uy4)
    vx=v[:,0].reshape((N,1))
    vy=v[:,1].reshape((N,1))

    constraint1_A = np.hstack([np.zeros((N,3)),ux,uy,np.ones((N,1)),-1*np.multiply(vy,ux),-1*np.multiply(vy,uy),-1*vy])
    constraint2_A = np.hstack([ux,uy,np.ones((N,1)),np.zeros((N,3)),-1*np.multiply(vx,ux),-1*np.multiply(uy,vx),-1*vx])
    A= np.vstack ([constraint1_A,constraint2_A])

    Create Jira Issue
    # TODO: 2.solve H with A
    U, S, VH = np.linalg.svd(A)

    # Let h be the last column of V , VH[-1,-1] = h9
    H=VH[-1,:]/VH[-1,-1]
    H=H.reshape((3,3))

    return H
```



Part2: Marker-Based Planar AR

- Paste the function code warping( ) (both forward & backward)



```
def warping(src, dst, H, ymin, ymax, xmin, xmax, direction='b'):
    h_src, w_src, ch = src.shape
    h_dst, w_dst, ch = dst.shape
    H_inv = np.linalg.inv(H)

    Create Jira Issue
    # TODO: 1.meshgrid the (x,y) coordinate pairs
    xx,yy=np.meshgrid(np.arange(xmin, xmax, 1), np.arange(ymin, ymax, 1))

    Create Jira Issue
    # TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate
    # [x1  x2  x3 ...]
    # [y1  y2  y3 ...]
    # [1   1   1  ...]
    x=np.expand_dims(xx.flatten(),0)
    y=np.expand_dims(yy.flatten(),0)
    one=np.ones((1,x.shape[1]))
    M = np.concatenate((x, y, one), axis = 0)

    # from given  (x,y) coordinate pairs destination(M)  to source
    if direction == 'b':
        Create Jira Issue
        # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        M_transformed  = np.dot(H_inv,M) # homogeneous coordinate
        M_transformed_Ordinary = np.round(np.divide(M_transformed[:2], M_transformed[-1,:])).astype(int)  # Ordinary Coordinate
        x_source=M_transformed_Ordinary[0].reshape((ymax-ymin, xmax-xmin))
        y_source=M_transformed_Ordinary[1].reshape((ymax-ymin, xmax-xmin))

        Create Jira Issue
        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
         # generate 2D boolean array for height and width
        h_mask = (0<y_source)*(y_source<h_src)
        w_mask = (0<x_source)*(x_source<w_src)
        # if both truth :mask = truth
        mask   = h_mask*w_mask

        Create Jira Issue
        # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
        x_source_masked=x_source[mask]
        y_source_masked=y_source[mask]

        Create Jira Issue
        # TODO: 6. assign to destination image with proper masking
        dst[yy[mask],xx[mask]] = src[y_source_masked, x_source_masked]
        pass

    # from given  (x,y) coordinate pairs source(M) to  destination
    elif direction == 'f':
        Create Jira Issue
        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        M_transformed  = np.dot(H,M) # homogeneous coordinate
        M_transformed_Ordinary = (np.round(np.divide(M_transformed[:2], M_transformed[-1,:]))).astype(int)  # Ordinary Coordinate
        x_destination=M_transformed_Ordinary[0].reshape((ymax-ymin, xmax-xmin))
        y_destination=M_transformed_Ordinary[1].reshape((ymax-ymin, xmax-xmin))

        Create Jira Issue
        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
        Create Jira Issue
        # TODO: 5.filter the valid coordinates using previous obtained mask
        # if exceed the boundaries then clip them
        np.clip(x_destination, 0, dst.shape[1]-1)
        np.clip(y_destination, 0, dst.shape[0]-1)

        Create Jira Issue
        # TODO: 6. assign to destination image using advanced array indicing
        dst[y_destination, x_destination] = src
        pass

    return dst
```

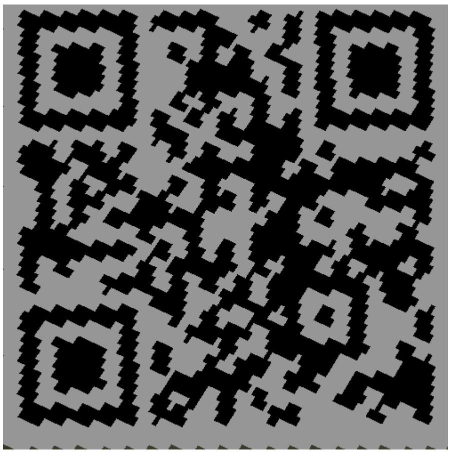
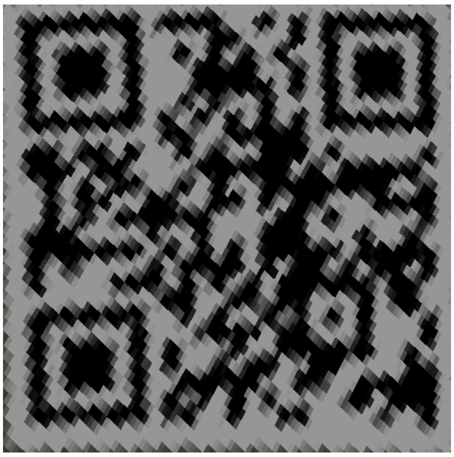
- Briefly introduce the interpolation method you use

  Forward 跟 backward 使用 Nearest neighbor Interpolation，齊次座標轉換過程中，用四捨五入(np.round)的方式找到最接近的整數值。

  另外，在 backward warping 的部分特別要注意超出 source 的部分，故要多一層 mask 先濾掉超出 source 的座標點再對照回去。

Part3: Unwarp the secret

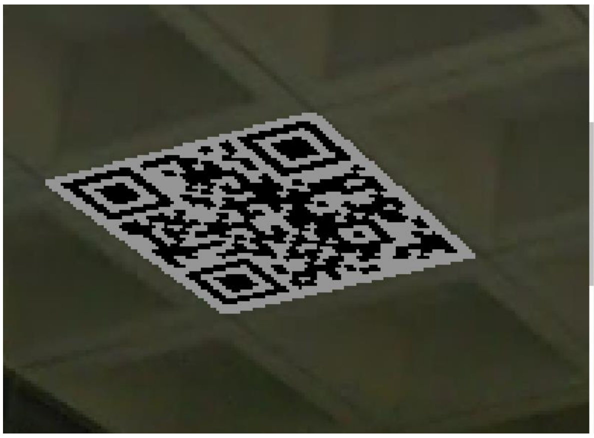
- Paste the 2 warped images

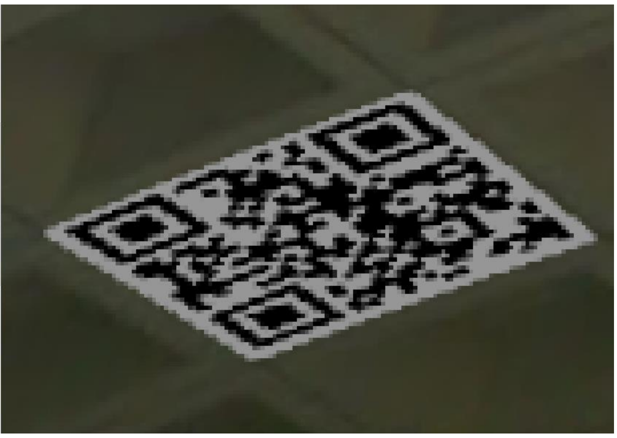
output3_1	output3_2
	

- Discuss the difference between 2 source images, are the warped results the same

or different? If the results are the same, explain why.If the results are different, explain why.

兩張照片即便是同一場所，但是拍攝到的線條很不一樣，BL_secret1 的 QRcode 是直線所組成的四邊形，BL_secret2 從相機拍的照片會有 Distortion，因此線條變成彎曲的樣子，細看發現 BL_secret2 原圖就有雜訊也比較模糊。



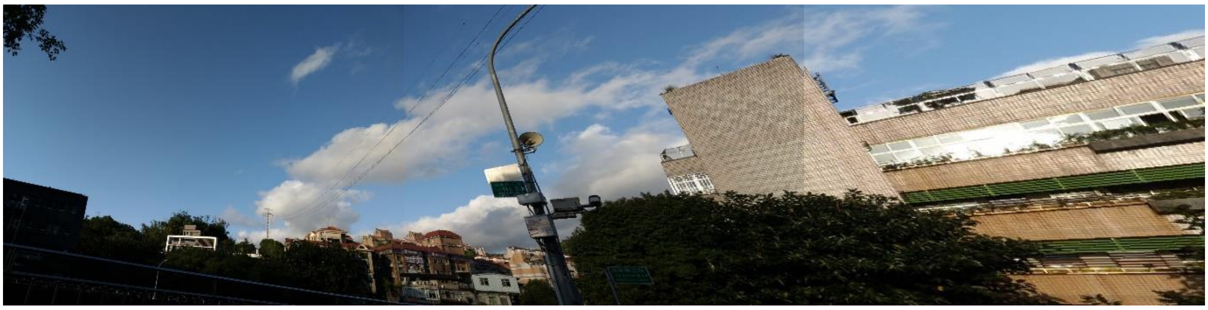


QRcode (都能掃到！) ：http://media.ee.ntu.edu.tw/courses/cv/21S/

warped results 有一點不一樣，BL_secret2 看的出來還有些微的扭曲，若有相機的相關參數，可以再有 Distortion Matrix 作校正。

Part4: Panorama

- Paste your stitched panorama



- Can all consecutive images be stitched into a panorama?

If yes, explain your reason. If not, explain under what conditions will result in a failure?

可以！大致可以看出來，不過還是會有連接不好的地方，如第二和第三張的顏色明顯有差異，且建築物的邊緣沒有完整接起來。

原因：

１、找feature的時候並沒有找出所有正確的keypoint（有錯誤）

２、RANSAC也只能找到「最適合」的H，只是逼近正確答案，終究有誤差

３、透過warping的時候還是需要內插法(nearest neighbor)，會有誤差

４、圖片的旋轉角不可以超過１８０度，以投影到平面來說，若兩張圖剛好差180度的話，兩個圖片將平行，沒有交點。